**UNIVERSIDAD AUTÓNOMA**

**“GABRIEL RENE MORENO”**

**FACULTAD DE CIENCIAS DE LA**

**COMPUTACIONY TELECOMUNICACIONES**

**

**Patrón Cadena de Responsabilidades**

**(Chain of Responsability)**

**Materia: Arquitectura de Software**

**Alumna: Dorado Medeiros Maria Nazarena**

**SANTA CRUZ 25-11-2019**

**Cadena de Responsabilidades**

**Área:**[Patrones de Diseño](http://www.juntadeandalucia.es/servicios/madeja/contenido/subsistemas/desarrollo/patrones-diseno)

**Carácter del recurso:**[Recomendado](http://www.juntadeandalucia.es/servicios/madeja/consultas/recursos?agrupaciones-caracter=8)

**Descripción**

El patrón de diseño Cadena de responsabilidades permite establecer una cadena de objetos receptores a través de los cuales se pasa una petición formulada por un objeto emisor. Cualquiera de los objetos receptores puede responder a la petición en función de un criterio establecido.

**Clasificación**

Patrón de comportamiento

**Aplicabilidad**

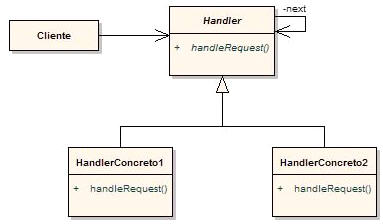
Más de un objeto necesita manejar una respuesta y el manejador no es conocido a priori.

Se quiere poder realizar una petición sin conocer a quién hay que solicitarlo.

El conjunto de objetos que pueden procesar una respuesta pueden ser especificados de forma dinámica.

**Estructura**

El patrón presenta la siguiente estructura:



**Participantes**

**Cliente**: Inicia la petición que llega a la cadena en busca del responsable.

**Manejador**: Define una interfaz para manejar peticiones.

**ManejadorConcreto**: Define las responsabilidades de cada componente. Si puede manejar una petición, la procesa, en caso contrario busca al siguiente.

**Colaboraciones**

Cuando un *Cliente* realiza una petición, está se propaga por el *Manejador* hasta que un *ManejadorConcreto* asume la responsabilidad de procesarla.

**Consecuencias**

***Añade Flexibilidad*** en la asignación de responsabilidades de los objetos. Se puede variar la asignación de responsabilidades añadiendo nuevos manejadores o modificándolos.

***La recepción no está asegurada***. Si la cadena no esté bien configurado puede no cubrir todas las peticiones.

***Reduce el acoplamiento***. El patrón permite que un objeto envié una petición y sepa que va a ser tratada correctamente, pero tanto el receptor como el emisor no conocen nada el uno del otro.

**Implementación**

Para implementar el patrón deben de tomarse las siguientes consideraciones:

1.-Implementar la cadena sucesora: Hay dos posibilidades de implementación. La primera es definir nuevos enlaces, ya sea en la clase *Manejador* o en la clase *ManejadorConcreto*. La segunda posibilidad es emplear enlaces ya existentes, por ejemplo empleando el patrón Composición.

2.-Conexión de los sucesores. Los propios *Manejadores Concretos* son los encargados de propagar incondicionalmente la petición. Las referencias deben de estar definidas.

3.-Representación de las peticiones. Se puede emplear el paso por parámetros o variables mediante una función manejadora, o hacer uso de invocaciones a operaciones insertadas en el código.